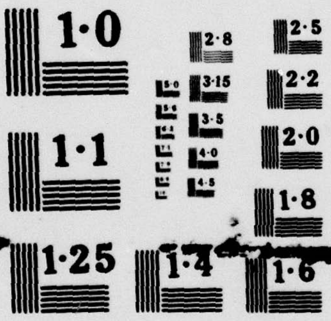
AD-A074 499 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 12/1
COMBINATORIAL SOLUTIONS OF MULTIDIMENSIONAL DIVIDE-AND-CONQUER --ETC(U)
MAY 79 L MONIER N00014-76-C-0370
UNCLASSIFIED CMU-CS-79-126 NL

| OF |
AD-
A074499

END
DATE
FILMED
11-79
DDC


1·0
2·8
2·5
3·15
2·2
5·0
3·5
2·0
1·1
4·0
4·5
1·8
1·25
1·4
1·6

Combinatorial Solutions of Multidimensional Divide-and-Conquer Recurrences

Louis Monier
IRIA-Laboria
78 Rocquencourt, France

[illegible] May 1979

DEPARTMENT
of
COMPUTER SCIENCE

CMU-CS-79-126

Combinatorial Solutions of Multidimensional Divide-and-Conquer Recurrences[1]

Louis Monier
Iria-Laboria
78 Rocquencourt, France

29 May 1979

Abstract

In this paper we use combinatorial techniques to solve recurrence relations in two variables of the form

$$T(N,k) = 2\,T(N/2,k) + T(N,k-1) + f(N)$$

and related recurrences. These recurrences arise in the analysis of algorithms based on a paradigm called "multidimensional divide-and-conquer". The analyses that we present are interesting from a combinatorial view, and show that certain algorithms are very efficient.

[1] This work was done while the author was visiting the Departement of Computer Science at Carnegie-Mellon University and was supported in part by the Office of Naval Research under Contract N00014-76-C-0370.

1. Introduction

In this paper we shall study the problem of (exactly) analyzing Multidimensional Divide-and-Conquer (MDC) algorithms described by Bentley [1978]. The execution costs of these algorithms are usually described by recurrence relations in two variables of the form

$$T(N,k) = a\ T(N/2,k) + b\ T(N,k-1) + f(N)$$

or

$$T(N,k) = a\ T(N/2,k) + b\ T(N/2,k-1) + f(N)$$

with initial values

$$T(1,k) = O(1) \text{ and } T(N,2) = g(N)$$

where a and b are integers, and f and g are functions of N. These recurrences have only been roughly solved for fixed k. The purpose of this paper is to solve these recurrences exactly using combinatorial techniques. The exact analysis gives us the *constant factor* in the expression of T(N,k) as a function of k, and also the ability to compare MDC algorithms with more obvious algorithms.

In Section 2 we sketch a particular MDC algorithm and derive the recurrence describing its running time. We solve that recurrence precisely in Section 3, and in Section 4 we use the combinatorial solution to describe the behavior of the algorithm. Section 5 is a collection of MDC recurrences and their solutions. In Section 6 we extend the solution method to more general recurrences, and conclusions are offered in Section 7.

2. The All-Points ECDF Algorithm

In this section we shall investigate a particular MDC algorithm and show how its recurrence can be derived; this algorithm is due to Bentley and Shamos and is described in Bentley [1978]. Our purpose in this section is not to learn all the details of the algorithm, but rather to understand how its recurrence arises. We say that a point $X=(x_1,\ldots,x_k)$ in an Euclidian k-space dominates point Y iff $x_i \geq y_i$ for all i. The rank r(X) of a point X is the number of points dominated by X. Given N points in k-space, the All-Points ECDF Problem is to compute the rank of each. The following is a sketch of Algorithm ECDFk described by Bentley [1978] for solving this problem on a set S of N points in k-space.

1. If the number of points, N, in S is one, then solve the problem in O(1) time; if the dimension, k, of the point set is two, then solve the problem in O(N lg N) time. If either of these conditions holds, return to the caller; otherwise continue to Step 2.

2. Using a hyperplane normal to one of the coordinate axes (say the x-axis), divide

S in two subsets A and B, each containing N/2 points.

3.Recursively solve the all-points ECDF problem on A and B (each problem of N/2 points in k-space).

4.Remarking that any point of B dominates each point of A in x-coordinate, we may remove this coordinate and solve the reduced problem (of finding for each point in B how many points of A it dominates) on the N points projected in (k-1)-space.

Let us denote by $T(N,k)$ the time for solving the all-points ECDF problem on a set of N points in k-space. Since Step 2 can be performed in time $\theta(N)$, we find the recurrence relation

$$T(N,k)=2\ T(N/2,k)+T(N,k-1)+\theta(N). \qquad (1)$$

From Step 1 we have the boundary conditions[2]

$$T(N,2)=\theta(N \log N)$$
$$T(1,k)=\theta(1)\ .$$

To analyze the above recurrence we must remove the "thetas". We will therefore solve the related recurrence

$$T(N,k)=2\ T(N/2,k)+T(N,k-1)+N \qquad (2)$$
$$T(N,2)=N \log N$$
$$T(1,k)=1.$$

The solution of the Equation 2 is always within a constant factor of solution of Equation 1, and therefore is precise enough for our purposes.

3. Combinatorial Solution

We shall first solve Equation 2 assuming that $N = 2^p$. For this purpose, we define

$$B(p,k)=T(2^p,k)/2^p\ .$$

Dividing both sides of Equation 2 by N and substituting this definition of B gives the reduced system

$$B(p,k)=B(p-1,k)+B(p,k-1)+1 \qquad (3)$$
$$B(p,2)=p$$
$$B(0,k)=1.$$

We may easily verify that the solution to Equation 3 is

[2]Here lg N denotes always $\log_2$ N.

$$B(p,k) = 2\binom{p+k-2}{k-2} + \binom{p+k-3}{k-1} - 1. \tag{4}$$

The solution of Equation 2 is now immediate; it is

$$T(N,k) = N\left[2\binom{\lg N + (k-2)}{k-2} + \binom{\lg N + (k-3)}{k-1} - 1\right] \tag{5}$$

We know now <u>what</u> the solution is; we do not, however, know <u>why</u> it is so. To achieve a more intuitive understanding of this solution, we shall transform Equation 3 by defining $A(p,k)=B(p,k)+1$ and find a combinatorial interpretation of the new recurrence

$$\begin{aligned} &A(p,k)=A(p-1,k)+A(p,k-1) \\ &A(p,2)=p+1 \\ &A(0,k)=2. \end{aligned} \tag{6}$$

Let us consider the net of Figure 1. The number $W_{(a,b)}(p,k)$ of ways *from* (a,b) to (p,k) using only edges in the network satisfies the recurrence

$$\begin{aligned} &W_{a,b}(p,k)=W_{a,b}(p-1,k)+W_{a,b}(p,k-1) \\ &W_{a,b}(p,0)=1 \\ &W_{a,b}(0,k)=1 \end{aligned}$$

which is the same as Recurrence 6, except for the bounds. Clearly, to go from (a,b) to (p,k) we must choose $(p-a)$ horizontal steps from a total of $(p-a)+(k-b)$ steps (the other $k-b$ being vertical). This immediately yields

$$W_{a,b}(p,k) = \binom{p-a+k-b}{k-a}.$$

We therefore build the net corresponding to Equation 6 and expand it on the line $k=1$; this is depicted in Figure 2. It is now easy to interpret $A(p,k)$ in terms of counting paths, and we find that

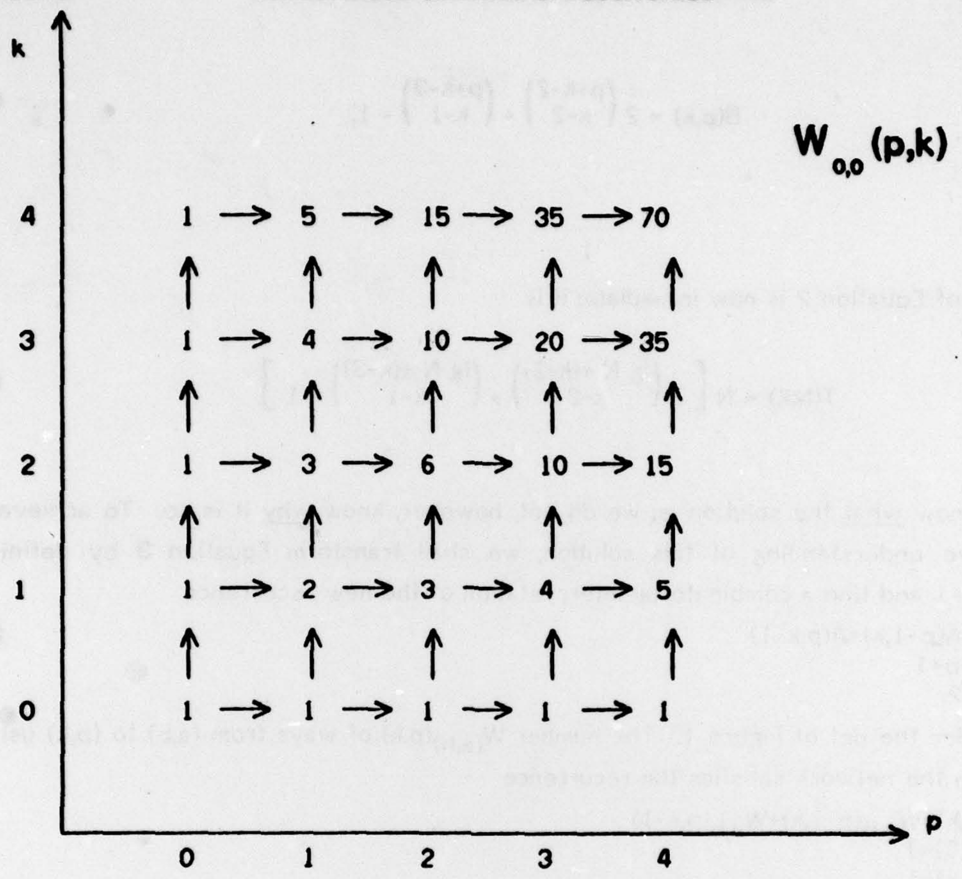
$$A(p,k) = 2\,W_{0,2}(p,k) + W_{2,1}(p,k)$$

which yields Equation 4.

In the next section, we shall study several other recurrences using this method, which we can summarize as follows.

-Manipulate the initial recurrence until obtaining a form similar to Equation 6.

-Build the corresponding net, and (if possible) expand it to make the bounds



$$\begin{tikzcd}
1 \arrow[r] & 5 \arrow[r] & 15 \arrow[r] & 35 \arrow[r] & 70 \\
1 \arrow[r] \arrow[u] & 4 \arrow[r] \arrow[u] & 10 \arrow[r] \arrow[u] & 20 \arrow[r] \arrow[u] & 35 \arrow[u] \\
1 \arrow[r] \arrow[u] & 3 \arrow[r] \arrow[u] & 6 \arrow[r] \arrow[u] & 10 \arrow[r] \arrow[u] & 15 \arrow[u] \\
1 \arrow[r] \arrow[u] & 2 \arrow[r] \arrow[u] & 3 \arrow[r] \arrow[u] & 4 \arrow[r] \arrow[u] & 5 \arrow[u] \\
1 \arrow[r] \arrow[u] & 1 \arrow[r] \arrow[u] & 1 \arrow[r] \arrow[u] & 1 \arrow[r] \arrow[u] & 1 \arrow[u]
\end{tikzcd}$$

Figure 1: A very simple net

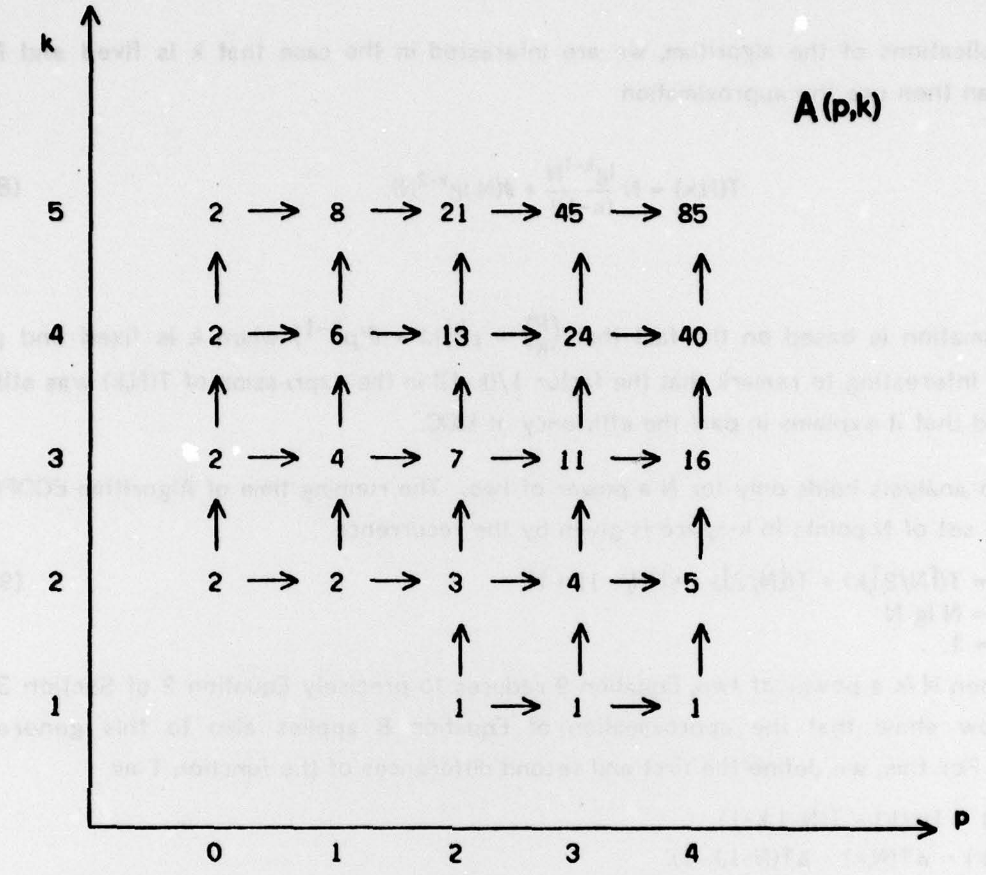


Figure 2: Net associated with Recurrence 6

"more simple".

-Interpret the recurrence in terms of paths and solve it.

4. Interpretation of the Solution

In the previous section we found that the running time of Algorithm ECDFk on a set of N points in k-space, for N a power of two, is given by the expression

$$T(N,k) = N\left[\,2\binom{\lg N + (k-2)}{k-2} + \binom{\lg N + (k-3)}{k-1} - 1\,\right] \qquad (7)$$

In this section we shall interpret this solution to see what it tell us about Algorithm ECDFk.

In many applications of the algorithm, we are interested in the case that k is fixed and N grows; we can then use the approximation

$$T(N,k) = N\frac{\lg^{k-1}N}{(k-1)!} + \theta(N \lg^{k-2}N). \tag{8}$$

This approximation is based on the fact that $\binom{p}{k} = p^k/k! + \theta(p^{k-1})$ when k is fixed and p grows. It is interesting to remark that the factor $1/(k-1)!$ in the expression of T(N,k) was still unknown, and that it explains in part the efficiency of MDC.

The above analysis holds only for N a power of two. The running time of Algorithm ECDFk on a general set of N points in k-space is given by the recurrence

$$\begin{aligned} T(N,k) &= T(\lceil N/2\rceil,k) + T(\lfloor N/2\rfloor,k) + T(N,k-1) + N \\ T(N,2) &= N \lg N \\ T(1,k) &= 1. \end{aligned} \tag{9}$$

Note that when N is a power of two, Equation 9 reduces to precisely Equation 2 of Section 3. We shall now show that the approximation of Equation 8 applies also to this general recurrence. For this, we define the first and second differences of the function T as

$$\begin{aligned} \Delta T(N,k) &= T(N,k) - T(N-1,k-1) \\ \Delta^2 T(N,k) &= \Delta T(N,k) - \Delta T(N-1,k-1). \end{aligned}$$

Using the recurrence defining the function T, it is easy to find the recurrence relations defining the differences. It becomes

$$\begin{aligned} \Delta T(N,k) &= \Delta T(N,k-1) + \Delta T(\lceil N/2\rceil,k) + 1 & \\ \Delta^2 T(N,k) &= \Delta^2 T(N,k-1) & \text{for even } N \\ \Delta^2 T(N,k) &= \Delta^2 T(N,k-1) + \Delta T(\lceil N/2\rceil,k) & \text{for odd } N \end{aligned}$$

We shall prove by induction on k and N that $\Delta T(N,k)$ is positive for all $N \geq 1$ and $k \geq 2$. For $k=2$ we verify this easily for all N since the function N lg N increases with N. When $N=2$, we have

$$\Delta T(2,k) = T(2,k) - T(1,k) = T(1,k) + T(2,k-1) + 1 > 0.$$

Assume that the first difference is positive in dimension k-1 for any N. Then a sufficient condition for $\Delta T(N,k)$ to be positive is that $\Delta T(\lceil N/2\rceil,k)$ is positive, and so on until we need $\Delta T(2,k)$ to be positive, which is true. We have therefore proved that for any k the function T(N,k) increases with N.

Using the same argument it is easy to prove that the second difference is always positive, since N lg N is a concave function of N and $\Delta^2 T(2,k) = \Delta^2 T(2,2) > 0$. So T(N,k) is itself a concave function of N for any fixed k. The previous results will be sufficient to prove that Equation 8 holds for all N and for any fixed k. For this purpose, we shall find upper and lower bounds of T(N,k) which are both of the form of Equation 8. Assuming that $2^p < N < 2^{p+1}$, and using the concavity of the function T(N,k), we may bound T(N,k) by S and I, as shown in Figure 3. Both values are found using linear interpolations of T between consecutive powers of two. The slopes of these linear interpolations being very close, we find for S and I expressions which are equivalent to the first order to

$$S = I = N \frac{p^{k-1}}{(k-1)!} + \theta((N+2^p)\, p^{k-2})$$

and since $p < \lg N \leq p+1$, the expressions of S and I are identical to the approximation of T(N,k) in Equation 8. Hence we have proved that Equation 8 holds for any $N \geq 1$ and $k \geq 2$.

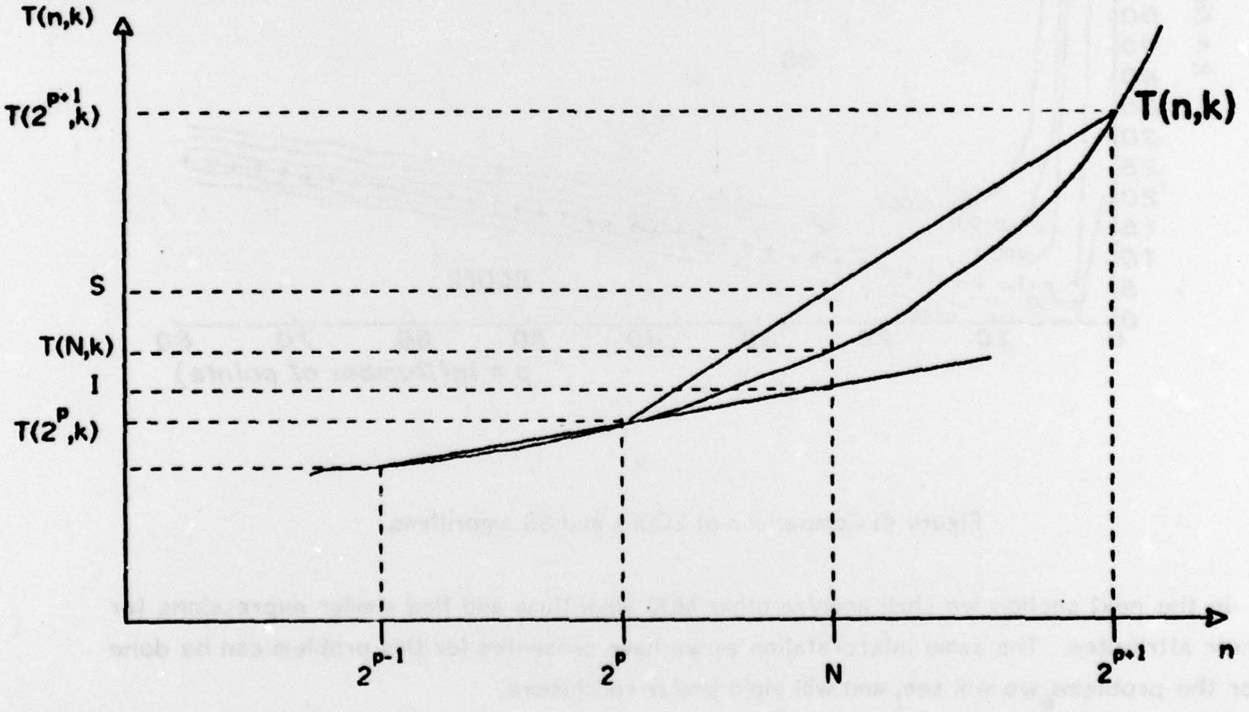


Figure 3: Approximation of T(N,k) using the concavity of the function.

This allows us to compare Algorithm ECDFk with the naive ECDF algorithm that compares all pairs of points, which we call the sequential searching (SS) algorithm. The complexity of this algorithm is $\theta(k\, N^2)$. For fixed k and large enough N it is clear that ECDFk is better than SS, since $\theta(N \lg^{k-1} N) < \theta(N^2)$. In some applications, however, the number of dimensions is too large to enable us to use the approximation of Equation 8. We then approximate the complexity of ECDFk by $c\,N\binom{\lg N+k}{k}$ and the complexity of SS by $k\,N^2$. Figure 4 shows in which domains each algorithm is faster, for various values of the parameter c. Since problems involving more than 2^{80} (or about 10^{24}) points will probably never be processed, Figure 4 covers the actual domain of values for k and

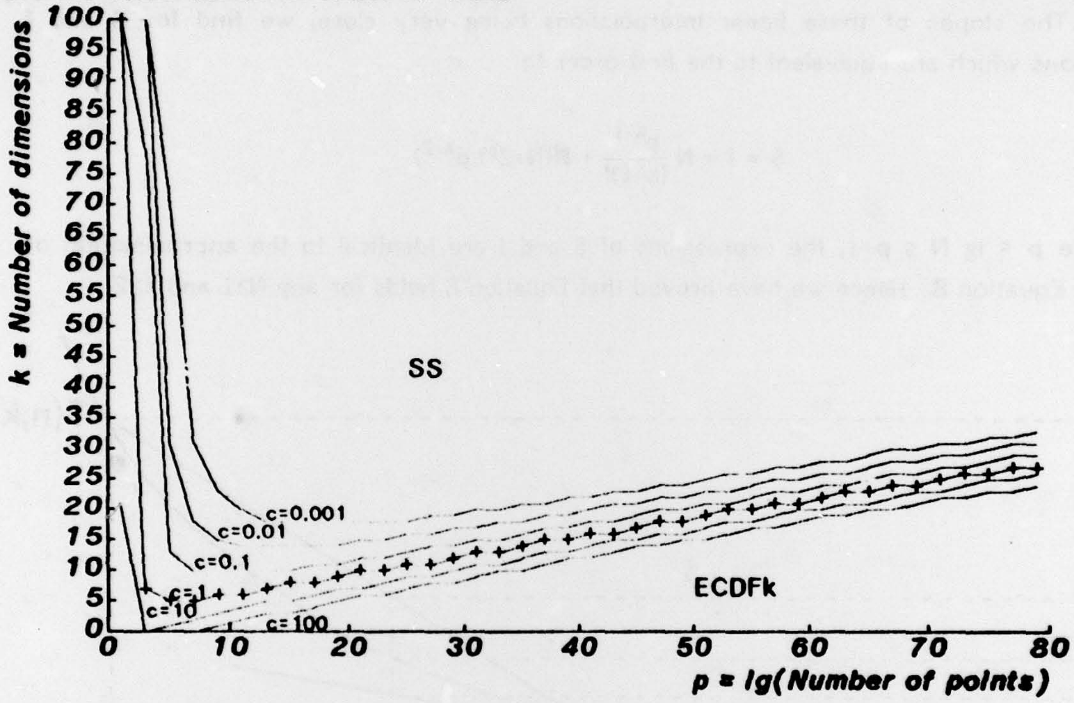


Figure 4: Comparison of ECDFk and SS algorithms.

In the next section we shall analyze other MDC algorithms and find similar expressions for their attributes. The same interpretation as we have presented for this problem can be done for the problems we will see, and will yield similar conclusions.

5. Examples

The following examples analyze the preprocessing time, storage and query time of two important algorithms, the Maxima Searching and the ECDF Searching Problems. A description of these algorithms may be found in Bentley [1978].

5.1. Maxima Searching

The first problem that we shall analyze is that of the *maxima searching* data structure. There are three attributes of this data structure to be analyzed: its preprocessing time (how long it takes to build the structure), its query time (how long it takes to search in the structure), and its storage (how much space is required to represent the structure). The first attribute that we shall analyze is the *preprocessing time*, which is given by the recurrence

$$P(N,k)=2\,P(N/2,k)+P(N,k-1)+\theta(N)$$
$$P(N,2)=\theta(N)$$
$$P(1,k)=\theta(1).$$

We can transform this recurrence using

$$B(p,k)=P(2^p,k)/2^p+1$$

into the following system

$$B(p,k)=B(p-1,k)+B(p,k-1)$$
$$B(p,2)=2$$
$$B(0,k)=2$$

which corresponds to a net similar to that of Figure 1 with somewhat different boundary conditions. The solution is

$$B(p,k)=2\binom{p+k-2}{k-2}$$

and we find the preprocessing time to be equal to

$$P(N,k)=2\,N\binom{\lg N+(k-2)}{k-2}-N.$$

For fixed k, as N grows, we may approximate P(N,k) by

$$P(N,k)=2\,N\frac{\lg^{k-2}N}{(k-2)!}+\theta(N\lg^{k-3}N)$$

We now turn our attention to the *storage* requirements, which are defined by

$$S(N,k)=2\,S(N/2,k)+S(N/2,k-1)$$
$$S(N,2)=N$$
$$S(1,k)=1.$$

We transform, as before, by

$$B(p,k)=S(2^p,k)$$

and we have to solve the system

$$B(p,k)=2\,B(p-1,k)+B(p-1,k-1) \qquad (10)$$
$$B(p,2)=2^p$$
$$B(0,k)=1.$$

The associated net is drawn in Figure 5.

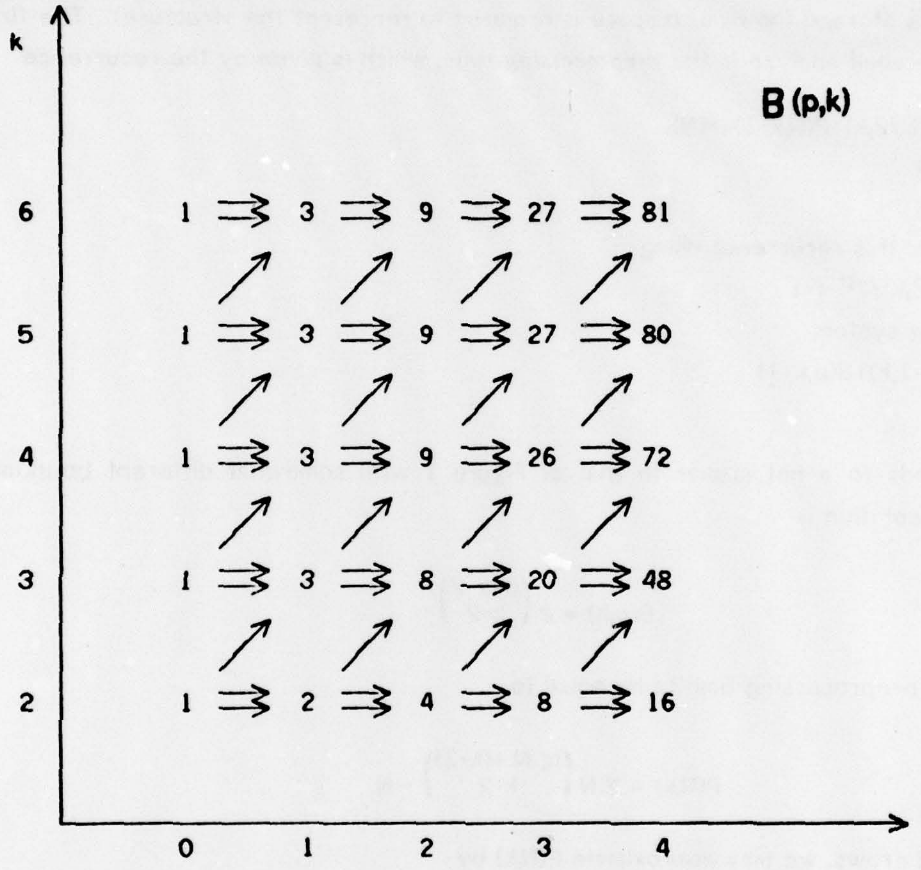


Figure 5: Net associated with Recurrence 10

Clearly, a way from (0,0) to (p,k) in such a net is a sequence of k diagonal steps and of p-k horizontal ones, each horizontal one chosen among two possibilities. So the number of ways from (0,0) to (p,k) is

$$2^{p-k}\binom{p}{k}.$$

Returning to the recurrence, we find that

$$B(p,k) = \sum_{2\le i\le k} 2^{p-(k-i)}\binom{p}{k-i}$$
$$= \sum_{0\le i\le k-2} 2^{p-i}\binom{p}{i}$$

and the storage is

$$S(N,k) = N \sum_{0\le i\le k-2} 2^{-i}\binom{\lg N}{i}.$$

We may remark that when $k \ge \lg N + 2$, the equation becomes

$$S(N,k) = \sum_{0\le i\le k-2} 2^{i}\binom{\lg N}{i} = 3^{\lg N} = N^{\lg 3}.$$

by the binomial theorem. An intuitive explanation of this fact is that for k large enough, we may equate S(N,k) and S(N,k-1) and write

$S(N,k) = 2\ S(N/2,k) + S(N/2,k-1) = 3\ S(n/2,k)$

whose solution is precisely $S(N,k) = 3^{\lg N} = N^{\lg 3}$.

At the opposite end of the spectrum, for fixed k and increasing N, we have

$$S(N,k) = N\frac{\lg^{k-2} N}{(k-2)!} + \theta(N \lg^{k-3} N).$$

The last attribute of the maxima searching data structure is the *query time*. Its associated system is

$Q(N,k)=Q(N/2,k)+Q(N/2,k-1)+\theta(1)$
$Q(N,2)=\lg N$
$Q(1,k)=1.$

The tranformation

$B(p,k)=Q(2^p,k)+1$

yields

$B(p,k)=B(p-1,k)+B(p-1,k-1)$
$B(p,2)=p+1$
$B(0,k)=2.$

The corresponding net is similar to Figure 5, but there is only one way for each diagonal step. So we find

$$B(p,k) = \binom{p-1}{k-1} + 2 \sum_{0 \le i \le k-2} \binom{p}{i}$$

and the query time is

$$Q(N,k) = \binom{\lg N - 1}{k-1} + 2 \sum_{0 \le i \le k-2} \binom{\lg N}{i} - 1$$

For fixed k, as N grows, we have

$$Q(N,k) = \frac{\lg^{k-1} N}{(k-1)!} + \theta(\lg^{k-2} N).$$

We must remark that the query time admits an interpretation similar to that of Section 4, and similar conclusion as for the comparison with the obvious SS algorithm. At the opposite, the storage and the preprocessing time are $\theta(N\ k)$ for SS, and hence better than those of the MDC Maxima Searching algorithms.

5.2. ECDF Searching

The second problem that we shall study is the ECDF searching; it is described by the same three attributes.

The *preprocessing time* and the *storage* are described by exactly the same recurrences, so we shall restrict our attention to the preprocessing given by

$P(N,k)=2\ P(N/2,k)+P(N/2,k-1)+\theta(N)$
$P(N,2)=N \lg N$
$P(1,k)=1.$

The function

$B(p,k)=P(2^p,k)+2^p$

gives us the new system

$B(p,k)=2\ B(p-1,k)+B(p-1,k-1)$
$B(p,2)=(p+1)\ 2^p$
$B(0,k)=2$

whose net is similar to Figure 5. We can expand the net by $B(p,1)=2^{p+1}$ for $p>0$. The solution is

$$B(p,k) = 2^{p-k+2}\binom{p-1}{k-1} + 2\sum_{0\le i\le k-2} 2^{p-i}\binom{p}{i}.$$

The preprocessing time and storage are given by

$$P(N,k) = N\left[\,2^{2-k}\binom{\lg N-1}{k-1} + 2\sum_{0\le i\le k-2} 2^{-i}\binom{\lg N}{i} - 1\,\right].$$

Note that if $k \le \lg N + 2$, then $P(N,k) = 2\,N^{\lg 3} - N$. For fixed k and large N the following approximation holds

$$P(N,k) = N\,\frac{\lg^{k-1} N}{(k-1)!} + O(N \lg^{k-2} N).$$

The last parameter to be analyzed is the *query time.* It is described by the system

$$Q(N,k)=Q(N/2,k)+Q(N/2,k-1)$$
$$Q(N,2)=\lg^2 N$$
$$Q(1,k)=1.$$

We use the new function

$$B(p,k)=Q(2^p,k)+1$$

and we have to solve

$$B(p,k)=B(p-1,k)+B(p-1,k-1)$$
$$B(p,2)=p^2+1$$
$$B(0,k)=2.$$

For this purpose, we expand the corresponding net by noticing that $B(p,1) = 2p+1$ and $B(p,0)=2$ are convenient for $p\ge 1$. It now becomes

$$B(p,k) = 3\binom{p-1}{k-1} + 2\binom{p-1}{k} + 2\sum_{2\le i\le k}\binom{p}{k-i}.$$

$$= 2\binom{p}{k} + \binom{p-1}{k-1} + 2\sum_{0\le i\le k-2}\binom{p}{i}$$

We substitute this expression in the initial equation to find

$$Q(N,k) = 2\binom{\lg N}{k} + \binom{\lg N-1}{k-1} + 2\sum_{0\le i\le k-2}\binom{\lg N}{i} - 1.$$

Again, we may remark that if $k \ge \lg N$, then $Q(N,k)=2^{\lg N} = N$. As usual, when k is fixed and N grows we find

$$Q(N,k) = 2\,\frac{\lg^k N}{k!} + \theta(\lg^{k-1} N).$$

6. More General Recurrences

We shall now study how we can solve the general reduced recurrence of the form

$$B(p,k) = a\,B(p-1,k) + b\,B(p-\epsilon,k-1)$$

where ϵ=0 or 1. The values of $B(p,k)$ are initially definite on a boundary E_0, and we assume that this bound is of the form of that in Figure 6. If this is necessary, we just consider a subset of E_0 of this form.

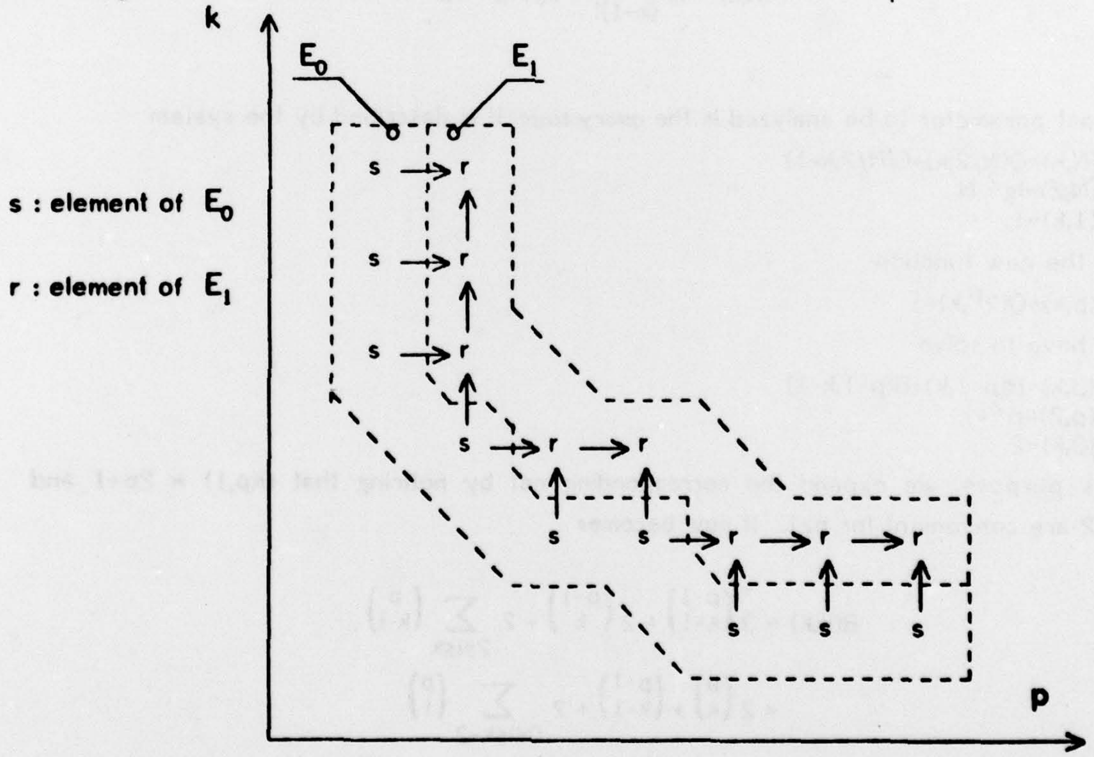


Figure 6: The sets E_0 and E_1 associated to a net.

We now define the set E_1 to be the set of points which are not in E_0 and which can be reached from E_0 in one step. Then, for any point (p,k) not in E_0, the function $B(p,k)$ is the sum of the number of ways from E_1 to (p,k), each way weighted by the value of B at its starting point $r_i=(p_i,k_i)$ in E_1. So we find the general solution

$$B(p,k) = \sum_{r_i \in E_1} W_{r_i}(p,k)\, B(p_i,k_i)$$

As in the Section 3, $W_{(r,s)}(p,k)$ denotes the number of ways from (r,s) to (p,k) in the current net. We may remark that the sum in the previous equation is finite, because $W_{(r,s)}(p,k)$ is null out of a finite sub-net, for fixed (p,k).

It may happen that the values of B are sufficiently regular on E_0 for allowing us to expand the net, and this sometimes reduces to a sum involving only a constant number of terms in the expression of B(p,k).[3]

Another problem is now to reduce the recurrences that appear in MDC-problems to the previous form. The general *paradigm* (divide N by 2 and solve the problem in a k-1 space) shows that the *natural* variables are lg N and k; so the primary change of variables is to use p = lg N. We therefore solve

$$B(p,k) = a\,B(p-1,k) + b\,B(p-\epsilon,k-1) + f(p,k)$$
$$B(p,k) = \text{known on a bound.}$$

The problem is to find a particuliar solution A(p,k), in order to solve the reduced equation verified by B-A. The only way seems to be luck and trick, and no general method can be exhibited here.

7. Conclusions

In this section we will briefly review the contributions of this paper. One of the main contributions has been the detailed analysis of the Multidimensional Divide-and-Conquer algorithms described by Bentley [1978]. We have exhibited precise analyses for many of those algorithms, accurate to within an (implementation-dependent) constant factor. These analyses show that the algorithms are more efficient than previously thought (it was not known that the constant of proportionality is the very small function 1/(k!)).

In addition to analyzing particular algorithms, we have seen a set of general tools applicable to the analysis of algorithms in two variables. The primary analytical tools are a set of useful transforms and an isomorphism of recurrences and path-counting problems on networks. We have also seen a number of tools for the interpretation of recurrences; these include a technique for showing the "smoothness" of the resulting function (between powers

[3] *For example, in the recurrences of Section 5*

of two) and a method for comparing sophisticated algorithms with more straightforward solutions. The methods that we have seen are applicable to all of the algorithms described by Bentley [1978], as well as other many others (such as in Lee and Wong [1979]).

8. Acknowledgements

At first, I should like to express my gratitude to Jon Bentley. By his help, advice and abundant encouragement he was a constant and efficient support. My thanks also to all the members of the Department of Computer Science at CMU who introduced me to the tools this paper has been prepared with, especially Don Cohen, Kevin Brown and Charles Leiserson. I want lastly to tell of my enjoyment for working in the CMU atmosphere.

References

Bentley, J.L. [1978] "Multidimensional Divide-and Conquer", to appear in *CACM*.

Lee, D.T. and Wong, C.K. [1979] "Quintary Trees: A File Stucture for Multi-dimensional Database System", to appear in *ACM Transactions on Database Systems*.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

14

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. ~~REPORT NUMBER~~ CMU-CS-79-126	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
6 4. TITLE (and Subtitle) COMBINATORIAL SOLUTIONS OF MULTIDIMENSIONAL DIVIDE-AND-CONQUER RECURRENCES -		5. TYPE OF REPORT & PERIOD COVERED 9 Interim rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 LOUIS/ MONIER		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-76-C-0370
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University Computer Science Department Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, VA 22217	11 29	12. ~~REPORT DATE~~ May [redacted]79
		13. ~~NUMBER OF PAGES~~ 18
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) 12 19p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

403 081 [illegible]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)